UNIVERSITY of ROCHESTER

Linear Filtering and Image Transforms

CSC 249/449 Spring 2019

http://www.cs.rochester.edu/~cxu22/t/249S19/

Instructor: Chenliang Xu
chenliang.xu@rochester.edu

Assignments and Python Tutorial

- Homework 1 is uploaded on Blackboard
- Due 1/31 midnight.

- We will have a Python tutorial Thu 1/24 7:30pm in 1400 Wegmans.

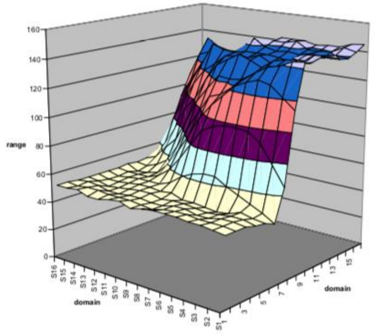
Today’s Content

- Images as functions
- Point operators
- Neighborhood operators
  - Linear filters
  - Nonlinear filters

(a) (b)

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

(c)

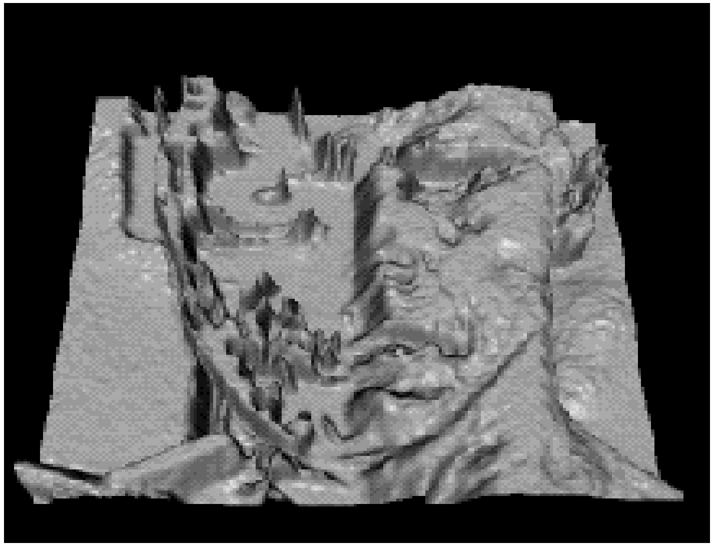


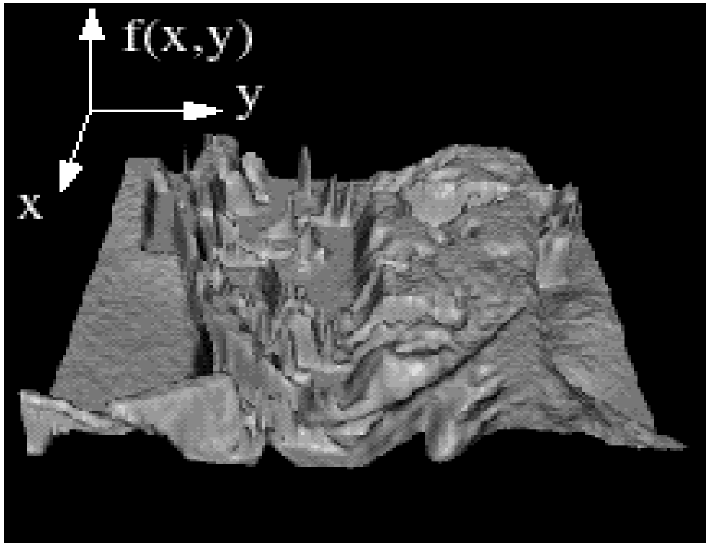
(d)

Figure 3.3 Visualizing image data: (a) original image; (b) cropped portion and scanline plot using an image inspection tool; (c) grid of numbers; (d) surface plot. For figures (c)–(d), the image was first converted to grayscale.

Figure from Szeliski

Images as functions

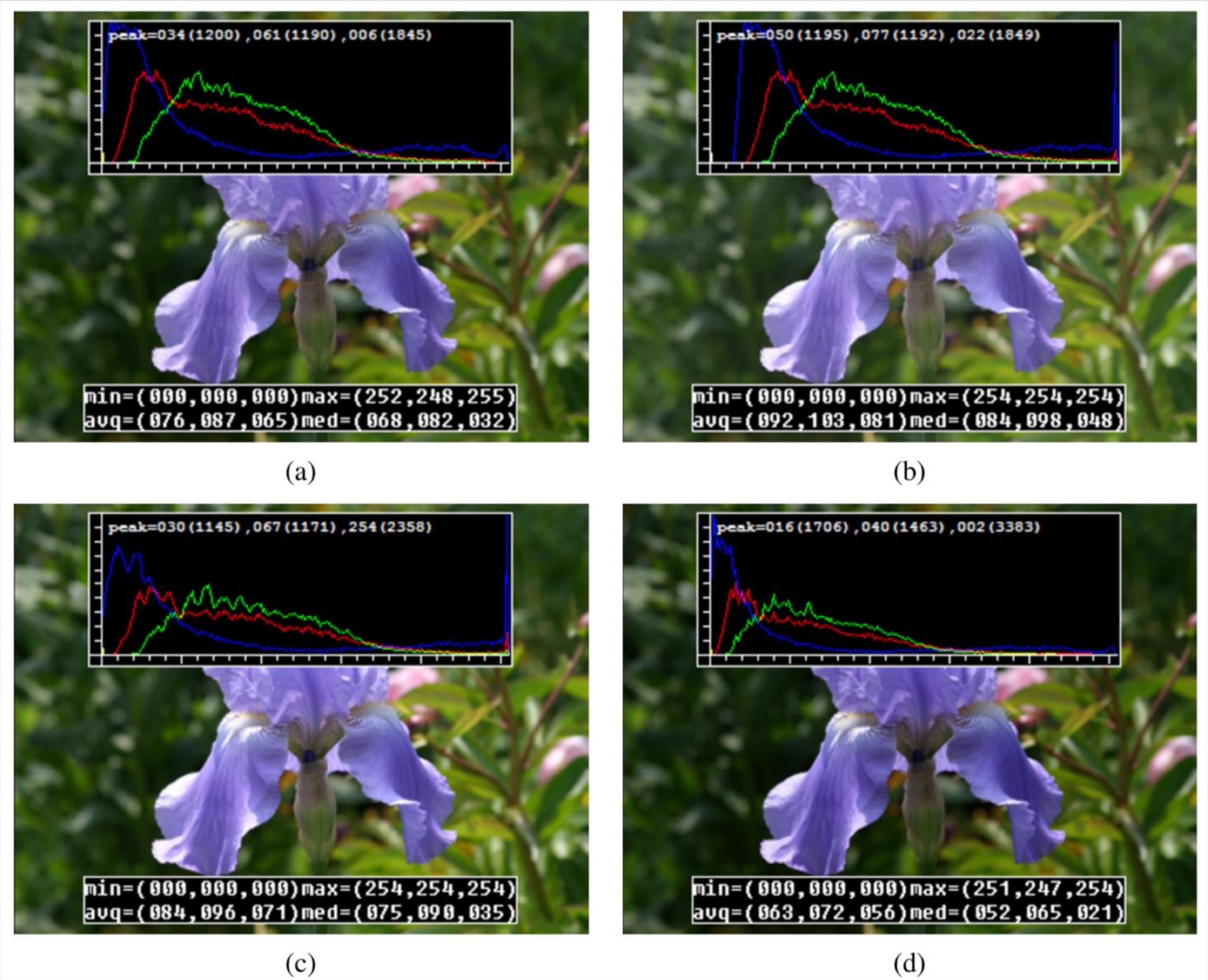




Adapted from: S. Seitz

Point Operators: Examples

Original Image | Brightness increased (additive offset, b=16)

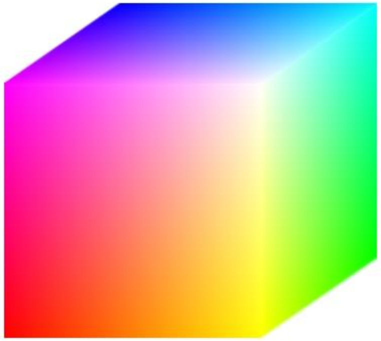


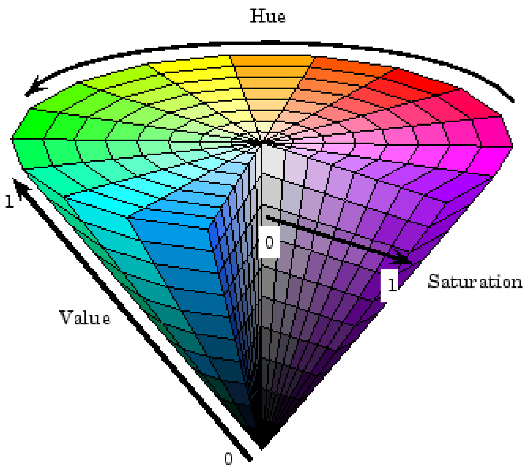
Contrast increased (multiplicative gain, a=1.1) | Gamma (partially) linearized (r=1.2)

Point Operators: Examples

- Color transforms

RGB primaries





RGB to HSV color table

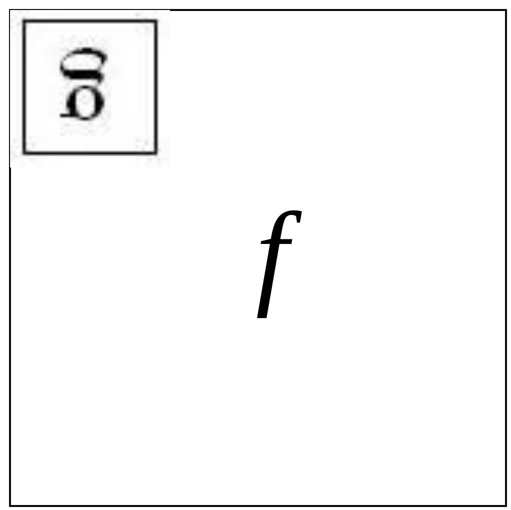
Color	Color name	Hex	(R,G,B)	(H,S,V)
	Black	#000000	(0,0,0)	(0°,0%,0%)
	White	#FFFFFF	(255,255,255)	(0°,0%,100%)
	Red	#FF0000	(255,0,0)	(0°,100%,100%)
	Lime	#00FF00	(0,255,0)	(120°,100%,100%)
	Blue	#0000FF	(0,0,255)	(240°,100%,100%)
	Yellow	#FFFF00	(255,255,0)	(60°,100%,100%)
	Cyan	#00FFFF	(0,255,255)	(180°,100%,100%)
	Magenta	#FF00FF	(255,0,255)	(300°,100%,100%)
	Silver	#C0C0C0	(192,192,192)	(0°,0%,75%)
	Gray	#808080	(128,128,128)	(0°,0%,50%)
	Maroon	#800000	(128,0,0)	(0°,100%,50%)
	Olive	#808000	(128,128,0)	(60°,100%,50%)
	Green	#008000	(0,128,0)	(120°,100%,50%)
	Purple	#800080	(128,0,128)	(300°,100%,50%)
	Teal	#008080	(0,128,128)	(180°,100%,50%)
	Navy	#000080	(0,0,128)	(240°,100%,50%)

Defining convolution

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted *f* * *g*.

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$

Convention:
kernel is "flipped"

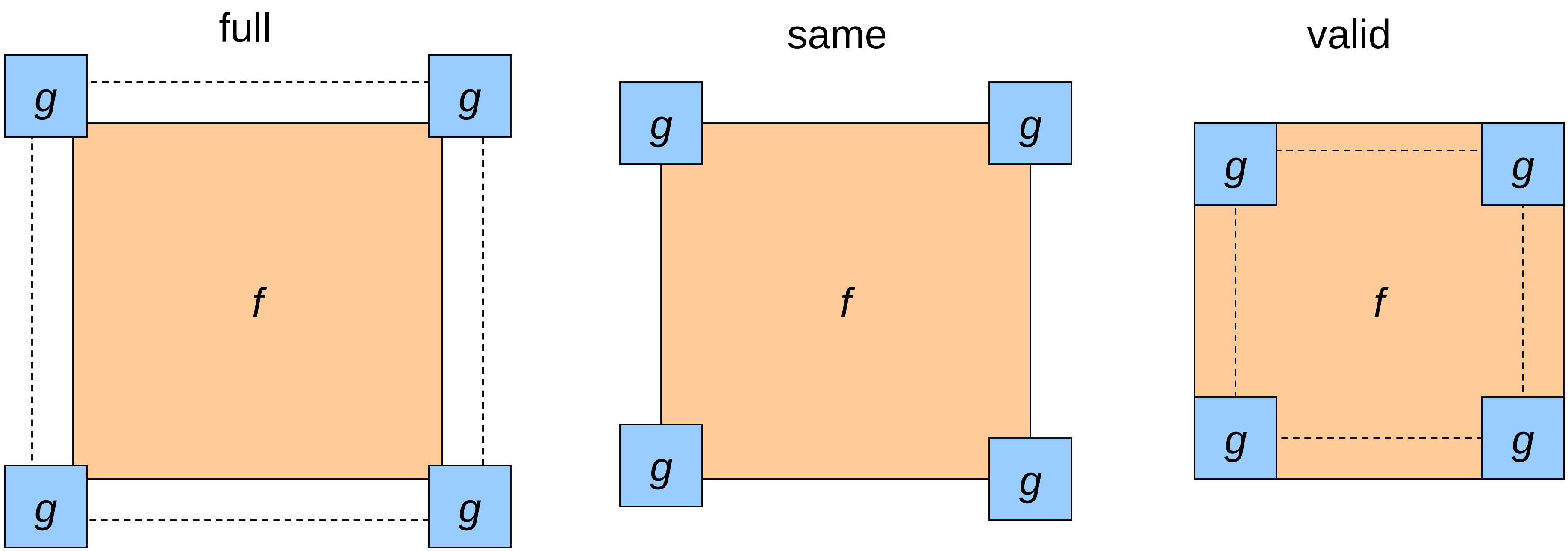


- MATLAB functions: conv2, filter2, imfilter

Source: F. Durand

Dealing with edges

- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Source: S. Lazebnik

Dealing with edges

- What about missing pixel values?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip black (zero-padding)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Dealing with edges

- What about missing pixel values?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods (MATLAB):
    - clip black (zero-padding): imfilter(f, g, 0)
    - wrap around: imfilter(f, g, ‘circular’)
    - copy edge: imfilter(f, g, ‘replicate’)
    - reflect across edge: imfilter(f, g, ‘symmetric’)

Source: S. Marschner

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

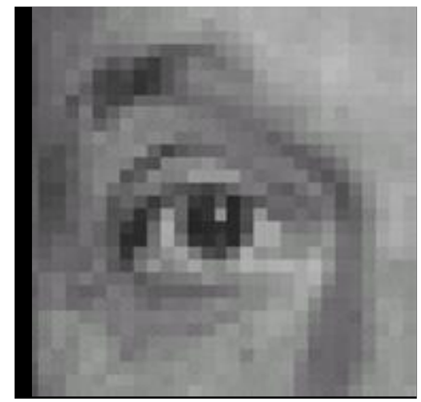
0	0	0
0	0	1
0	0	0

?

Source: D. Lowe

Practice with linear filters

0	0	0
0	0	1
0	0	0



Original

Shifted *right*
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

?

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Blur (with a
box filter)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

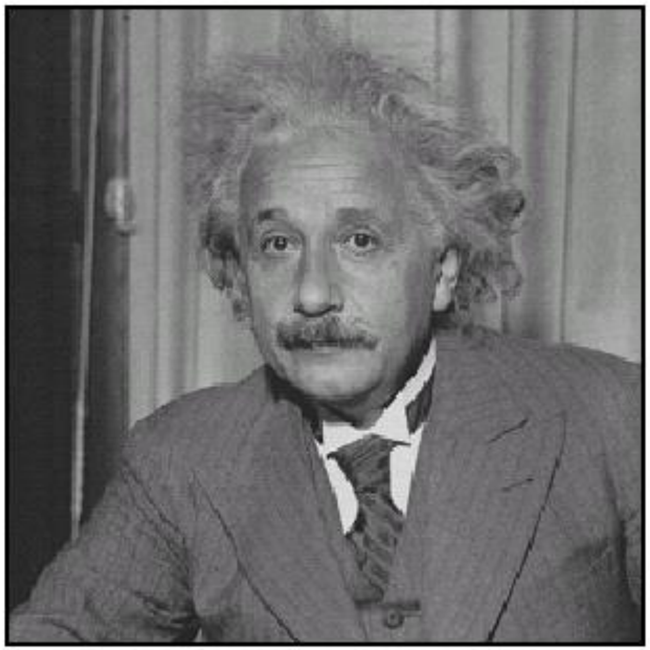
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

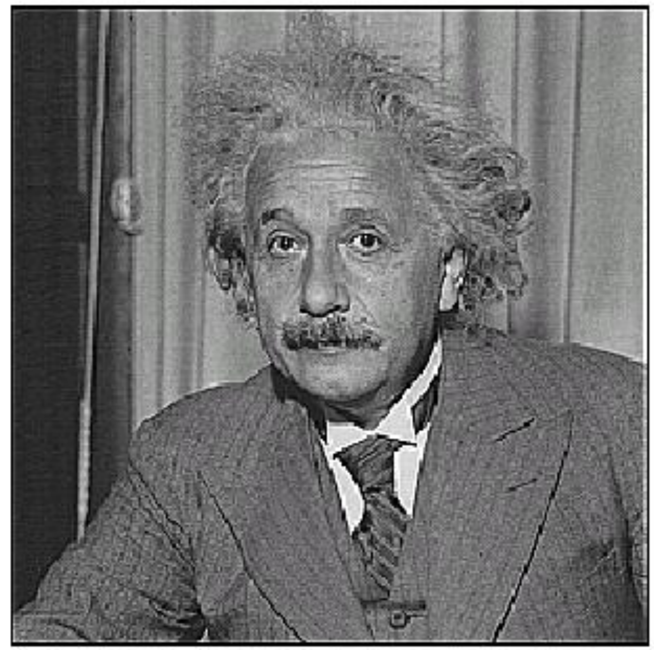
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before

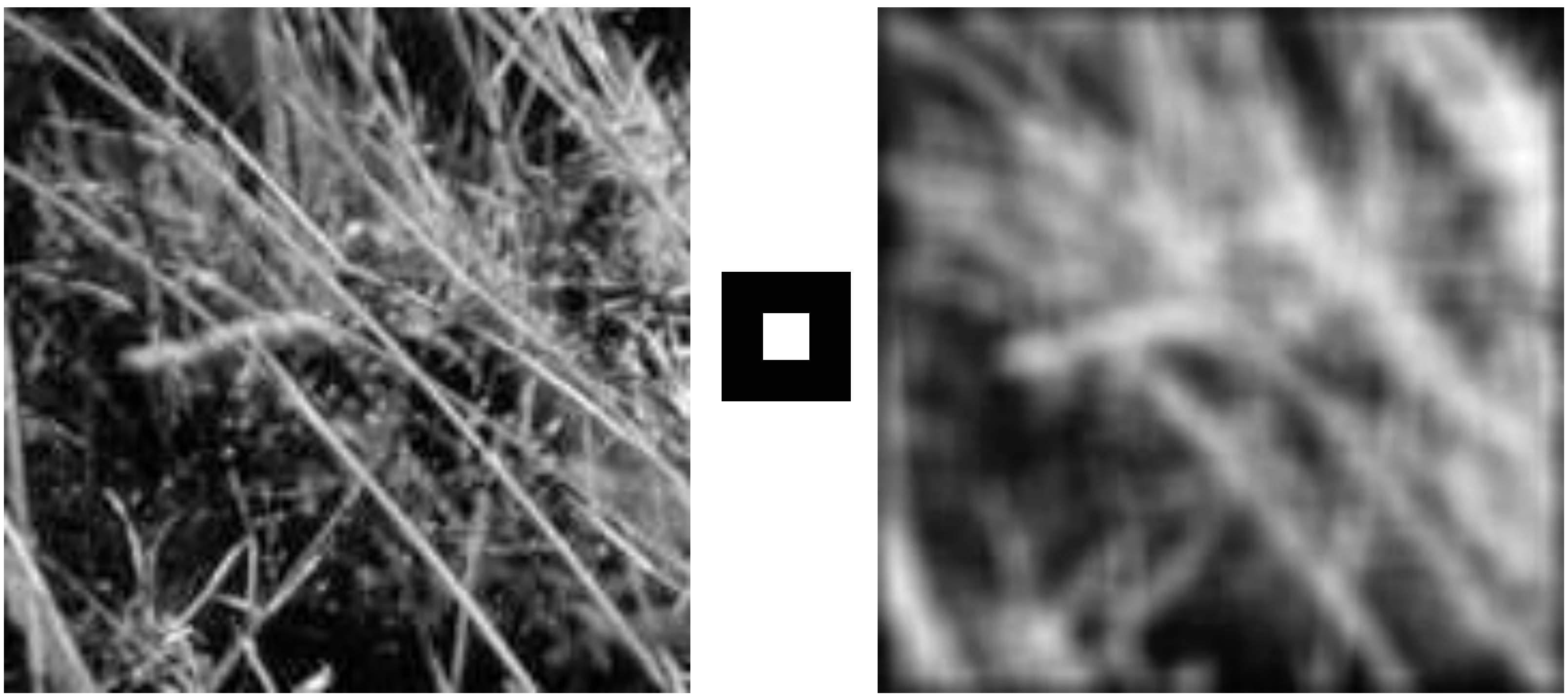


after

Source: D. Lowe

Smoothing with box filter revisited

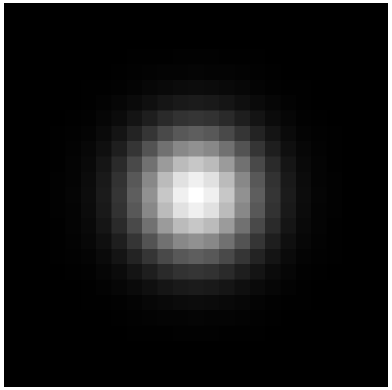
- What's wrong with this picture?
- What's the solution?



Source: D. Forsyth

Smoothing with box filter revisited

- What's wrong with this picture?
- What's the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

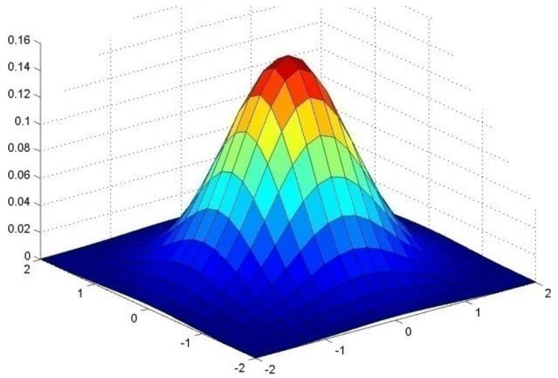


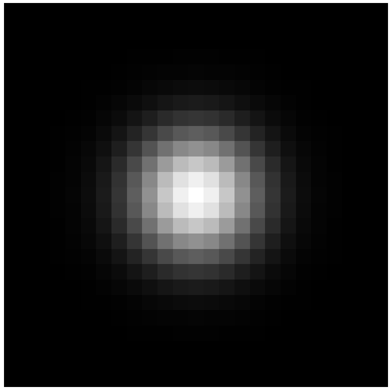
"fuzzy blob"

Source: S. Lazebnik

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

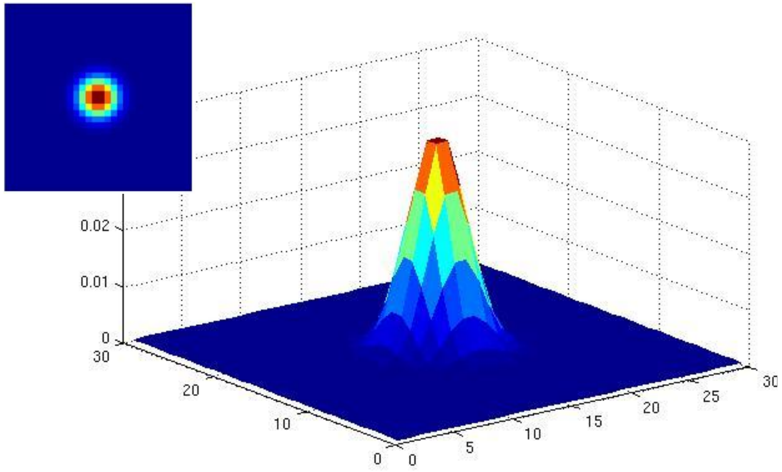
5 x 5, σ = 1

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

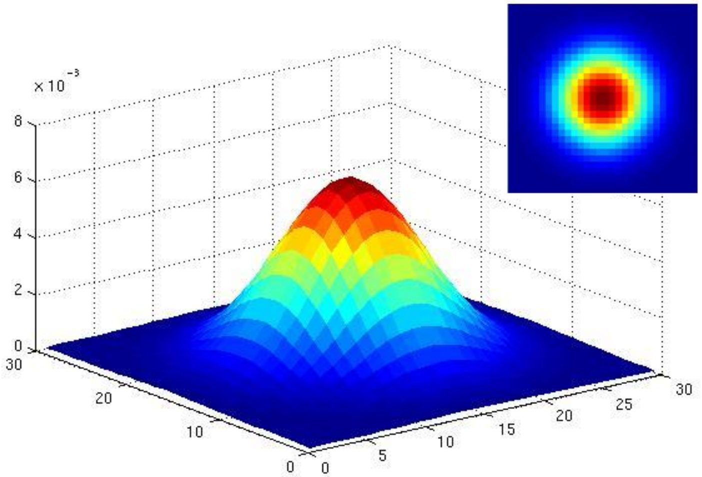
Source: C. Rasmussen

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



σ = 2 with 30 x 30 kernel



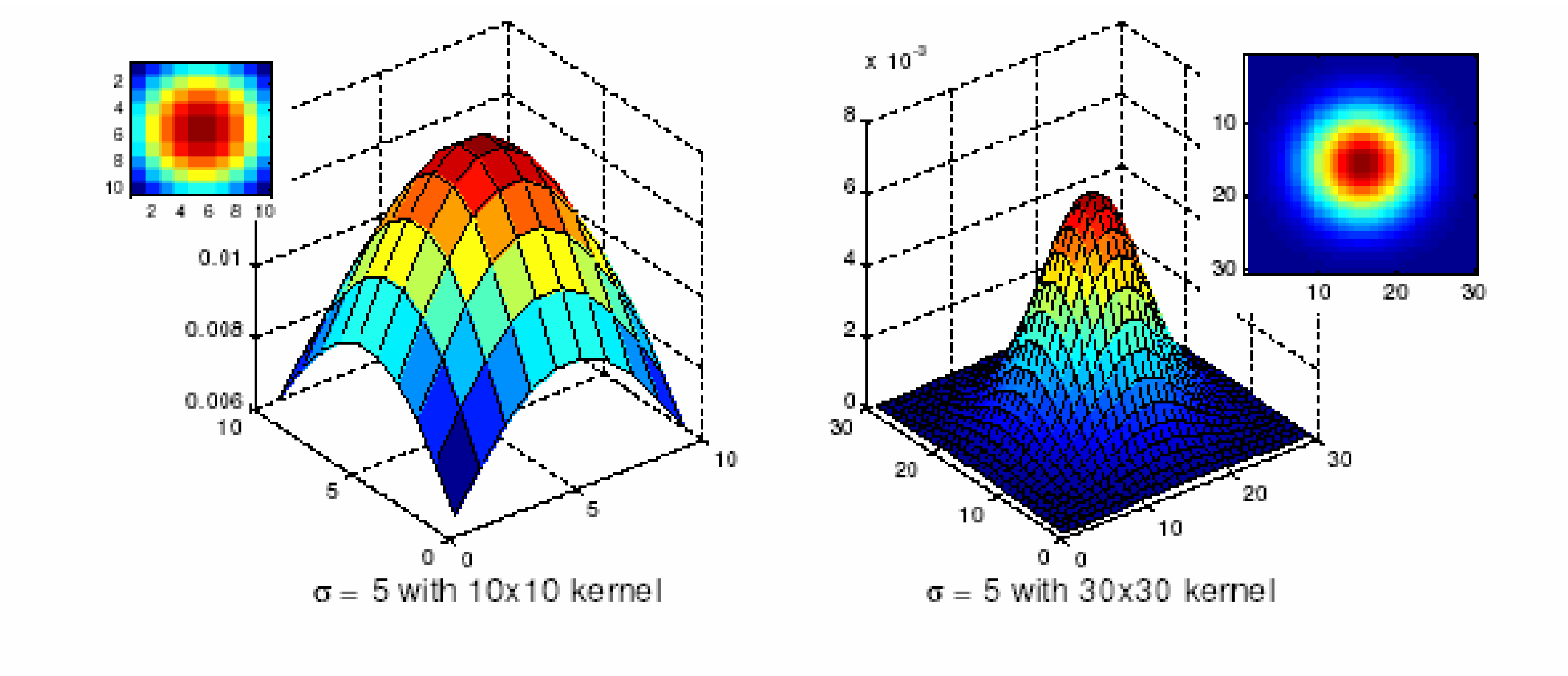
σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

Source: K. Grauman

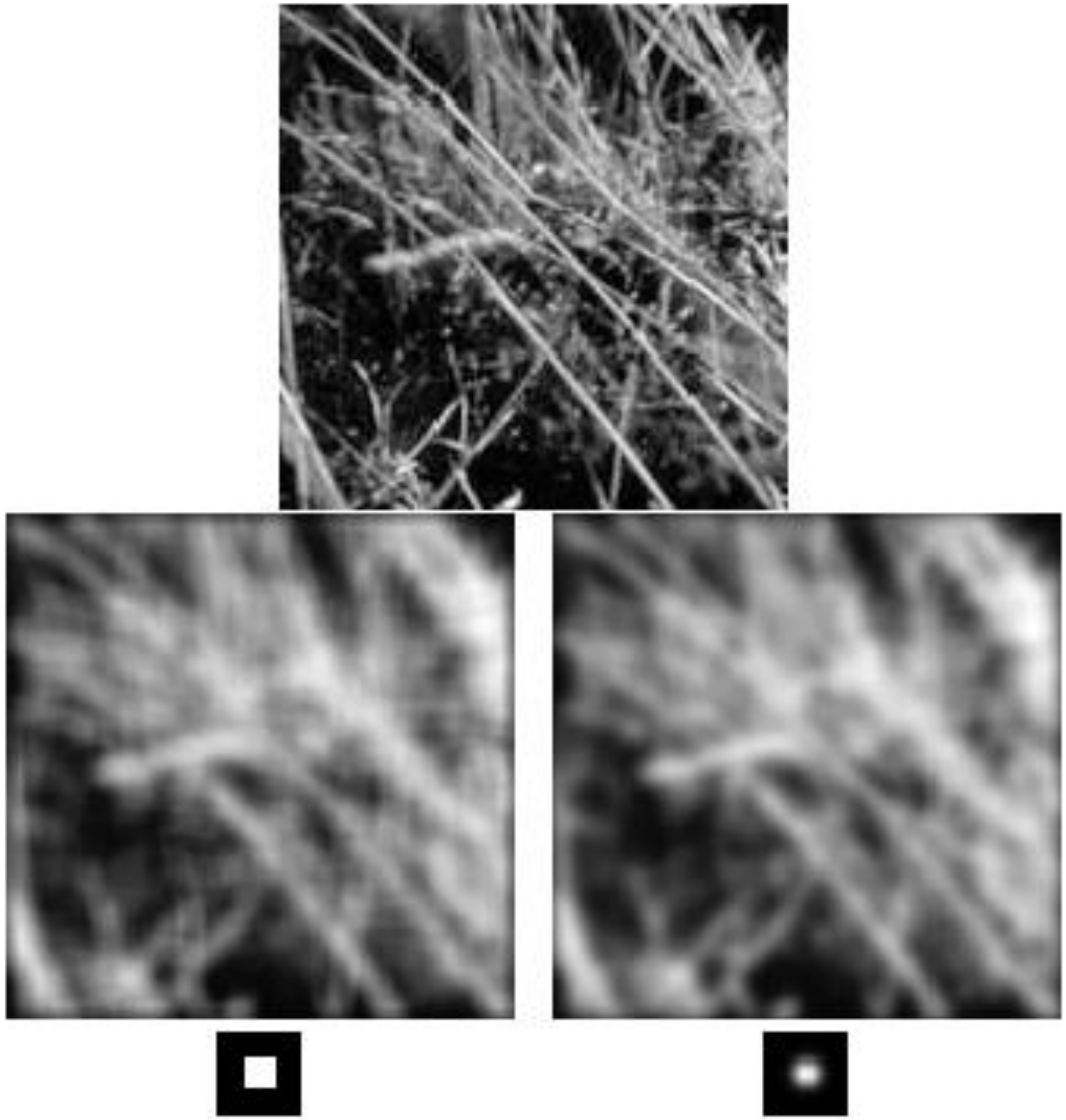
Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels



Source: K. Grauman

Gaussian vs. box filtering



Source: S. Lazebnik

Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Source: K. Grauman

Separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Non-linear Filters
(and denoising)

Noise

Original

Salt and pepper noise

Impulse noise

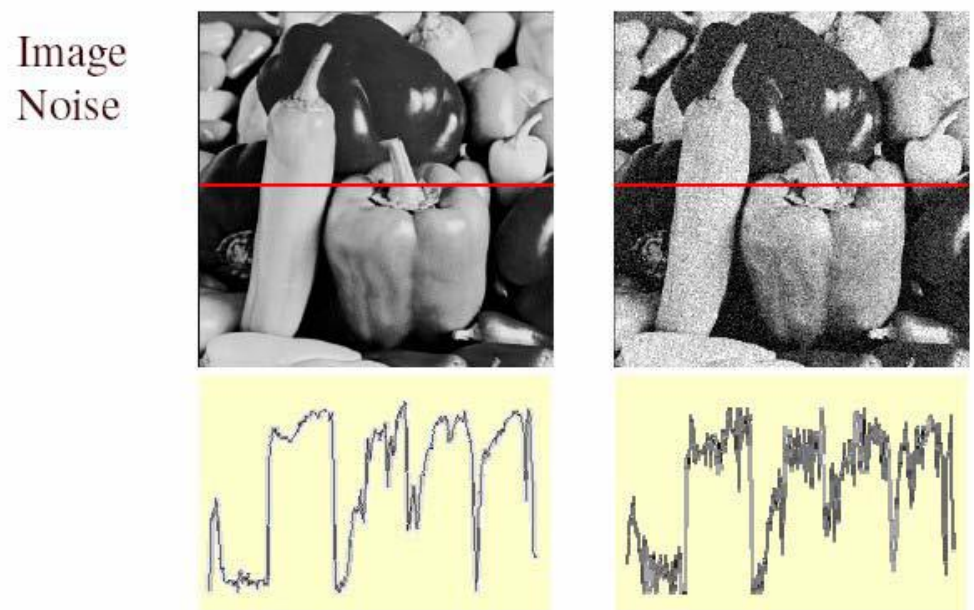
Gaussian noise

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

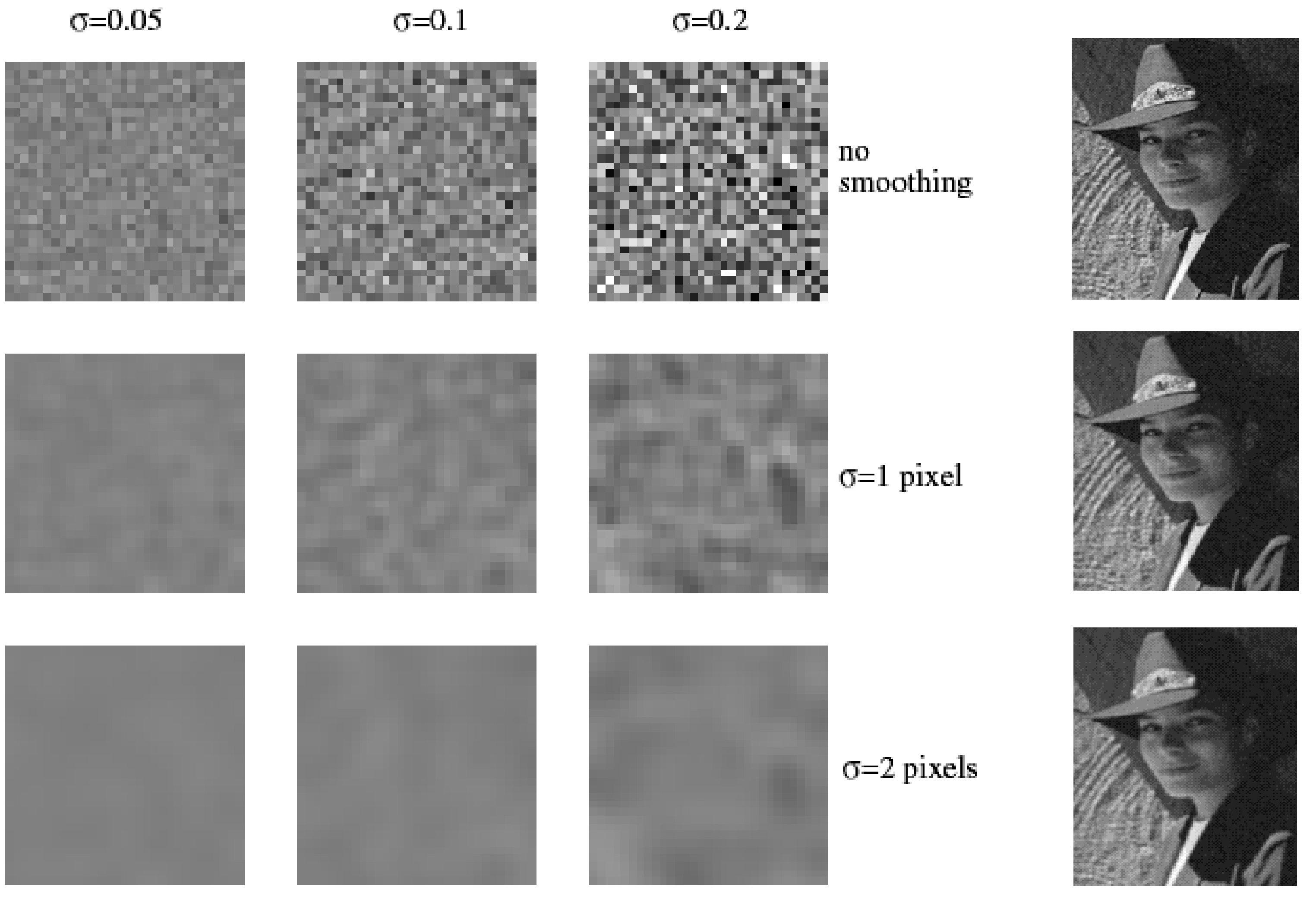


$$f(x,y) = \overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

Source: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Source: S. Lazebnik

Reducing salt-and-pepper noise

3x3

5x5

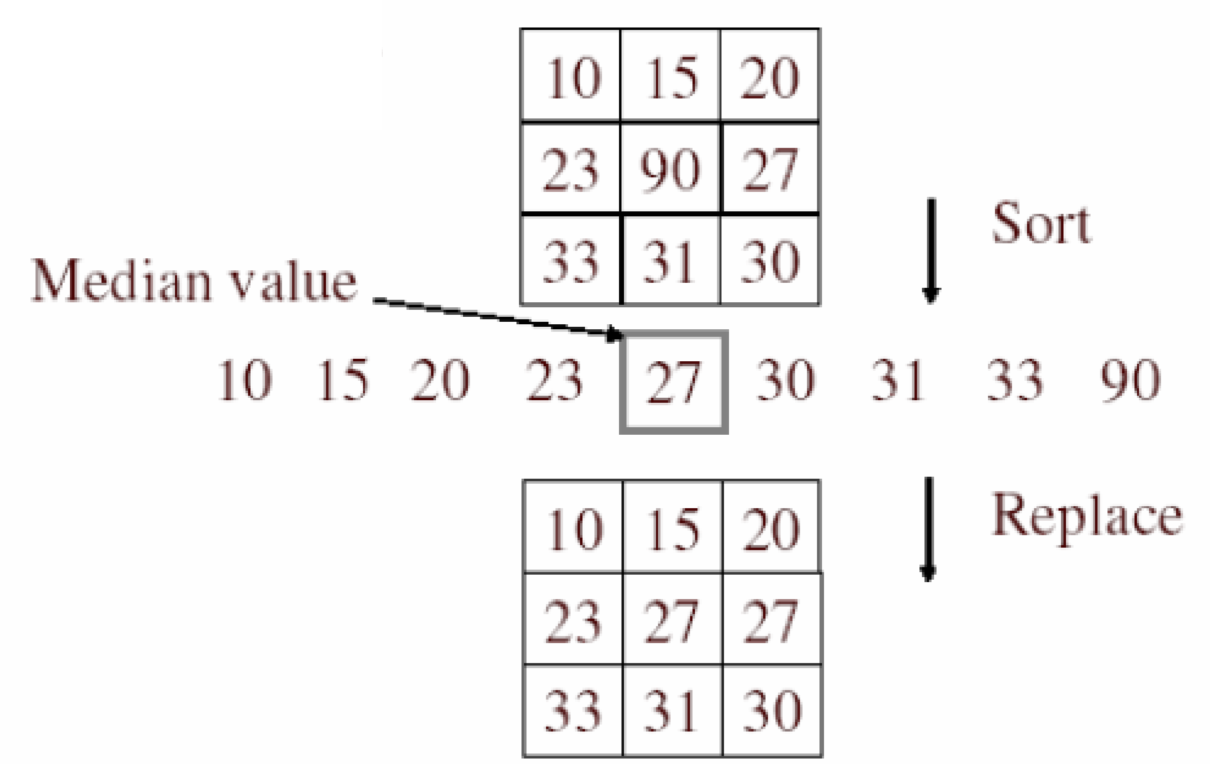
7x7

- What’s wrong with the results?

Source: S. Lazebnik

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window



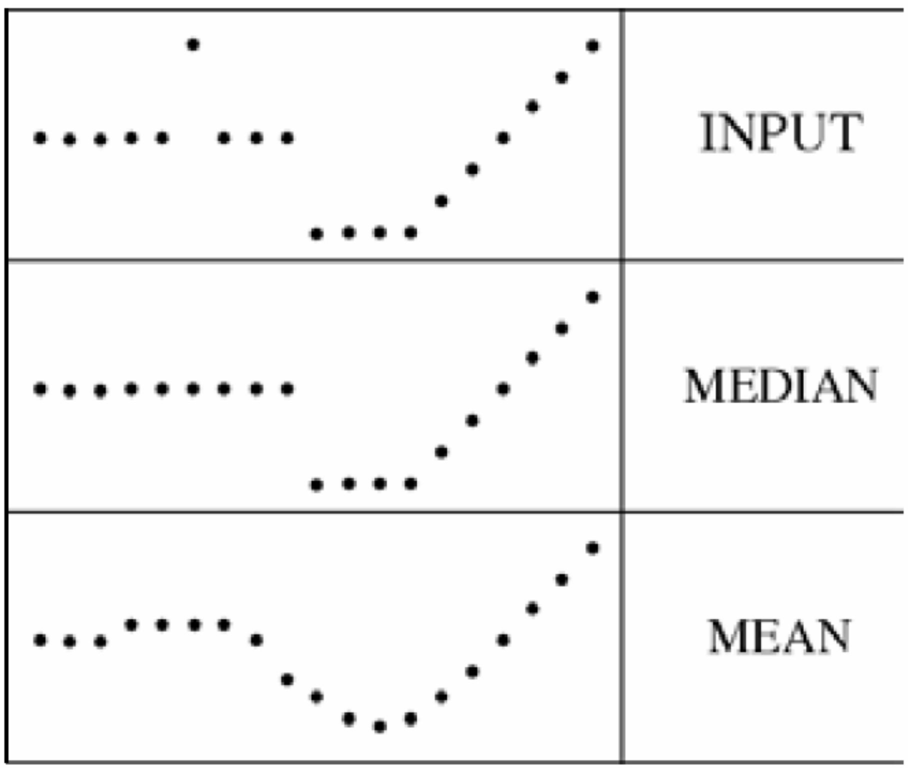
- Is median filtering linear?

Source: K. Grauman

Median filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers

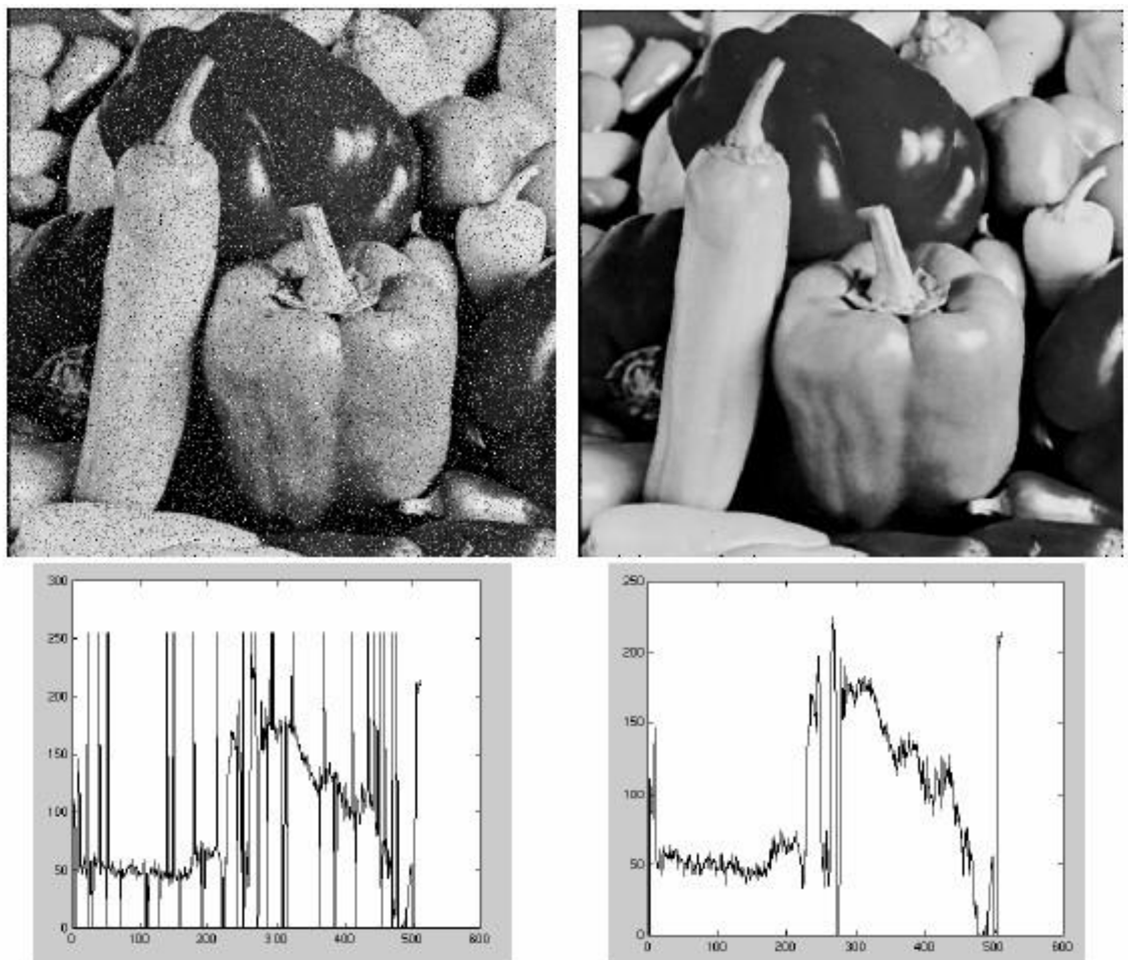
filters have width 5 :



Source: K. Grauman

Median filter

Salt-and-pepper noise Median filtered



- MATLAB: medfilt2(image, [h w])

Source: M. Hebert

Gaussian vs. median filtering

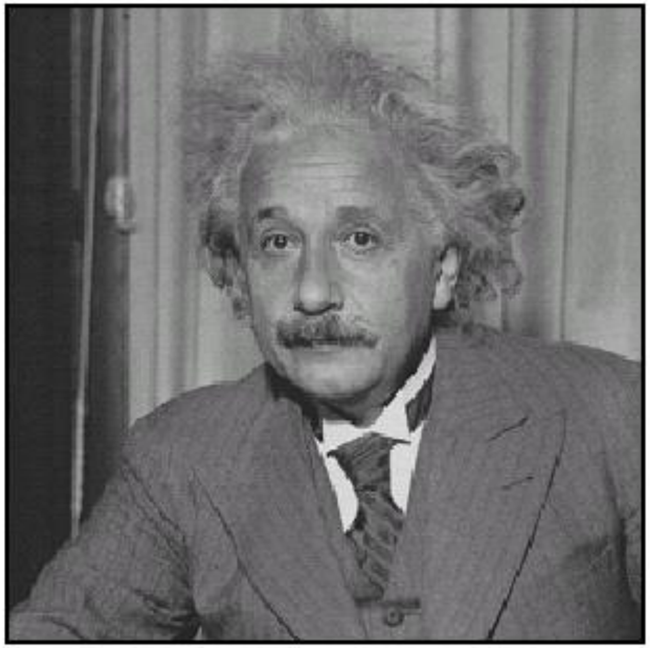
3x3 5x5 7x7

Gaussian

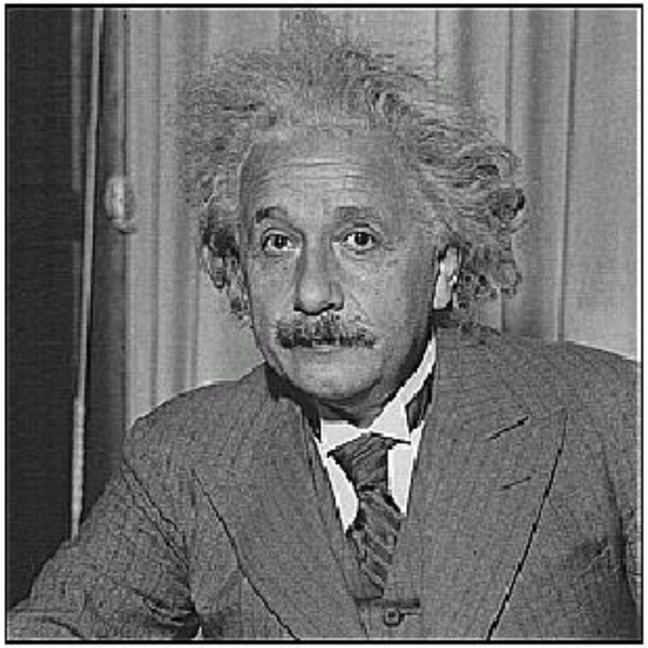
Median

Source: S. Lazebnik

Sharpening revisited



before



after

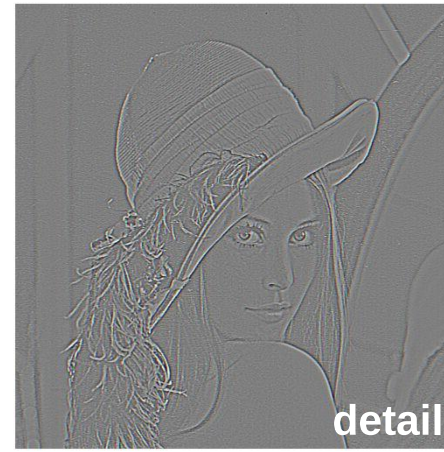
Source: D. Lowe

Sharpening revisited

- What does blurring take away?

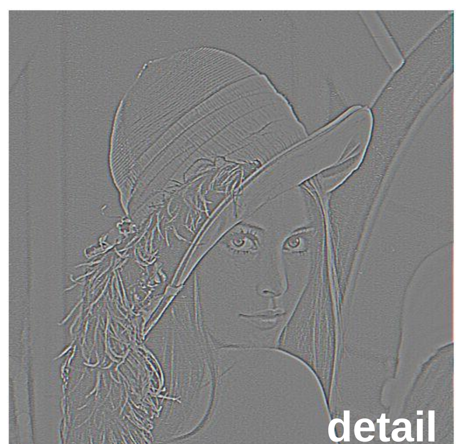
–

=



Let's add it back:

+ α



=

Source: S. Lazebnik

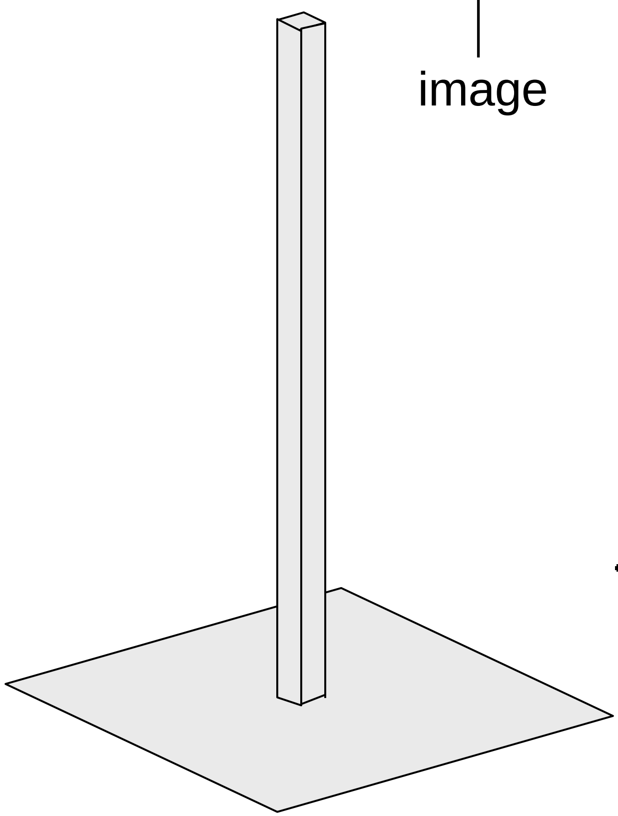
Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha) f - \alpha f * g = f * ((1+\alpha)e - g)$$

image

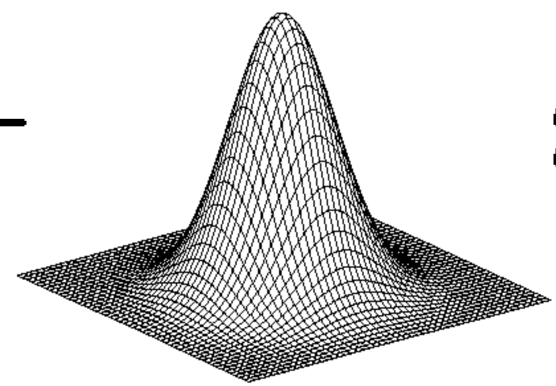
blurred image

unit impulse (identity)



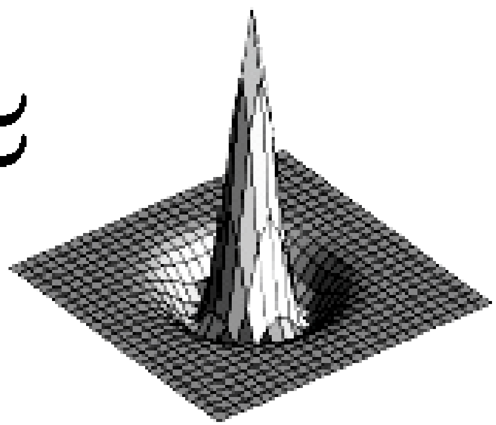
unit impulse

$-$



Gaussian

$\approx$



Laplacian of Gaussian

Source: S. Lazebnik

Application: Hybrid Images

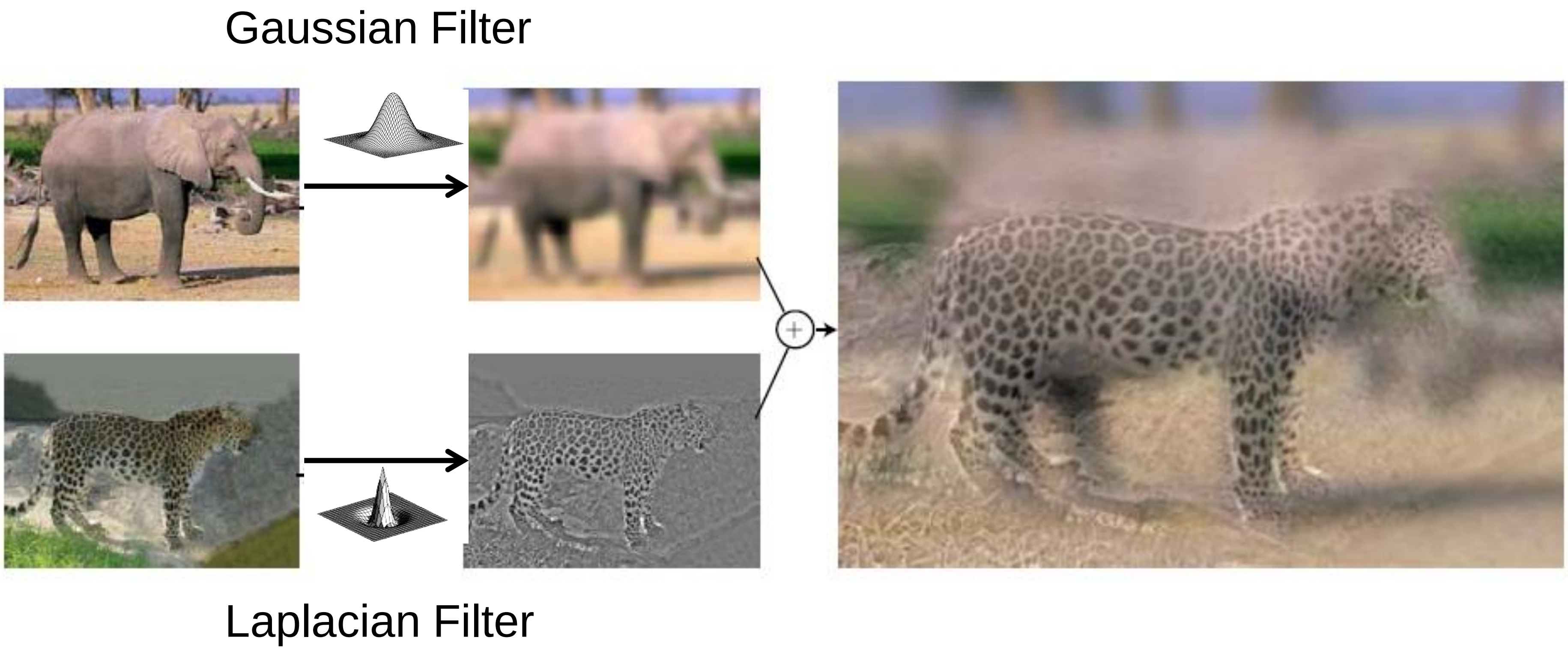
- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Changing expression

SIGGRAPH2006

Sad ⟷ Surprised

Application: Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Summary

- Images as functions
- Point operators
- Neighborhood operators
  - Linear filters
  - Nonlinear filters

- Optional Reading:
  - Fourier transforms (Images as points), Sec. 3.4.1-2

- Next Lecture:
  - Edges and Corners
  - Sec. 3.2.3, 4.2, 4.1.1 (Up to P. 190)